User Feedback Section

User Feedback

Overview

One of the principles of user centered design is to provide feedback at all times to the user. Feedback informs the user what is going on. Feedback can be visual or audio and, should be presented with every user interaction to confirm that the software is responding. Feedback needs to be informative, timely, and presented as close to the point of the user's interaction as possible.

Feedback can be as simple as a changing the mouse pointer, or more complex like warning and error messages that appear in message boxes. The key point is that the feedback needs to be appropriate to the type of task the user is performing. In addition, all CDC applications need to provide feedback in a consistent manner. This section attempts to define appropriate types of feedback for both Windows and web-based surveillance applications.

General Feedback Standards

The following feedback standards are to be adopted across all CDC Windows and web based surveillance applications. These standards should lead to a more consistent and usable interface.

Messaging

In Windows, most messages are communicated to the user with a message box. There are 4 types of message boxes, which are illustrated in the table titled *Message Box Types*. Each box has different implications and any inappropriate uses will be discussed in the general standards section of this document.

Message Box Types

Type	Description	Icon
Flash Box	Similar to notification except that the message box removes itself after a couple of seconds.	ℹ
Notification *(Informational)*	Provides the user with information about the results of a command. Offers no user choices. Implemented using the *Informational* icon.	ℹ
Confirmation *(Warning)*	Alerts the user to a condition that requires user input before proceeding. This message is often in the form of a question. Implemented with the *Warning* icon.	⚠
Error *(Critical)*	Informs the user of a serious problem that requires corrective action before work can continue. Implemented with the *Critical* icon.	✖

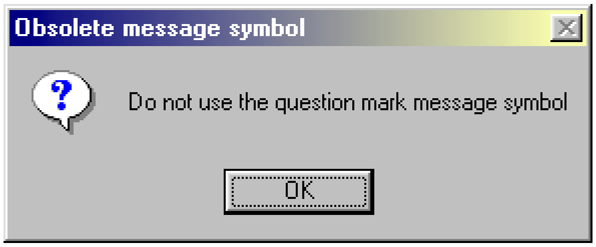
Show a single message box.

Show messages in one message box, not several consecutive message boxes.

Do not use the question mark as a symbol in a message box.

The question mark symbol has been deprecated. It was used in earlier versions of Windows for confirmation messages. This icon, as illustrated below in the figure titled *Inappropriate Message Symbol*, is no longer recommended because the question mark is now associated extensively with Help Information.

Inappropriate Message Symbol



Obsolete message symbol

Do not use the question mark message symbol

Use the message box title bar text to display application name.

Providing an appropriate identifier for messages is very important in the Windows multitasking environment, because message boxes might not always be the result of the current user interaction. The title bar communicates the source of the message. Do not use descriptive text for message box title text such as "warning" or "caution". The message symbol described above conveys the nature of the message. Never use the word error in the title text.

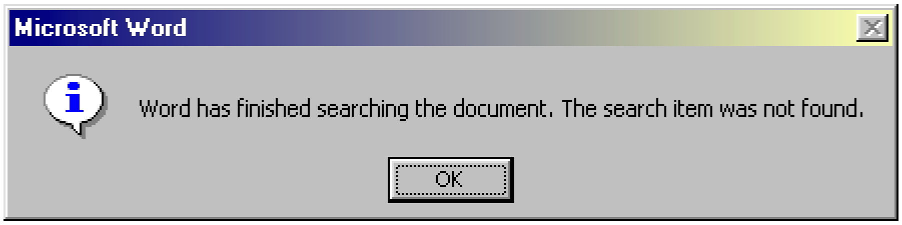
Do not use Flash Boxes.

Flash boxes are relatively new to Windows and are not often used. Since they stay visible for only a few seconds, the user may miss the message. Do not use flash message boxes in CDC software.

Replace notification message boxes with status bar messages whenever possible.

A notification message, also called an informational message, can be eliminated and moved to the status bar if the information it provides is not significant or could be replaced with some other type of visual feedback. Notification message boxes should only be used when the successful completion of a command is not obvious and there is no alternative way to provide the feedback. The figure titled *Notification Messaging* illustrates how Microsoft Word uses a notification message box to let the user know that there are no more occurrences of the specified text in a search. The status bar was not used in this case because of the risk that the user might make several passes through the document without realizing the search had been completed.

Notification Messaging



Word has finished searching the document. The search item was not found.

Use the status bar for contextual user assistance.

Consider status bar messages as a secondary or supplemental form of user assistance. Because the status bar location may not be near the user's area of interaction, the user may not always notice a status bar message. Appropriate uses of the status bar messages include::

- *Instructions or messages that do not fit easily into a ToolTip*
- *Descriptive messages about menus and menu items*
- *Instructional messages for enterable field values*
- *Feedback on disabled commands or fields*

Use the status bar for processing feedback.

There are user-initiated processes, such as printing or file saving, and background processes, which require some feedback to let the user know that a program will not be able to respond. Use the status bar for these messages. In addition, it is helpful to display a "processing completed successfully" message to let the user know, for example, that a patient record has been added or updated successfully. In these messages be sure to include some piece of information (name or id) that identifies what has just been processed.

Minimize confirmation messages.

Any action that can easily be undone, should not be confirmed. Do not confirm a user action just because you think the user might make a mistake. This will lead to your program being sprinkled with unnecessary confirmations, which undermine the objective of protecting the user. Too many confirmations will result in users simply clicking OK without reading the message.

Provide confirmation messages when exiting a program only if changes may be lost.

Users should be able to exit a program without always being asked if they are sure that they want to exit. The only time the user should be asked for confirmation when exiting is when they have changes that have not been saved and those changes would be lost should they exit.

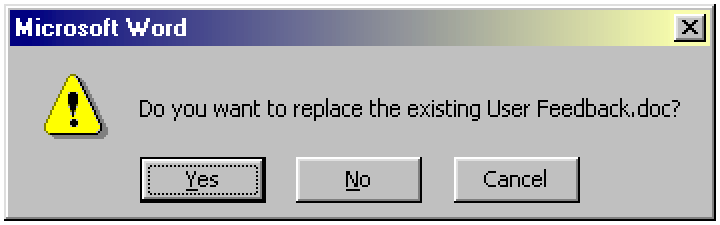
Do not confirm to save changes unless changes have actually been made.

Never ask the user if they want to save changes unless they have explicitly made changes.

Always confirm destructive irreversible actions with a question to the user.

When there is an impending action that has destructive and irreversible consequences, such as deleting a record or a file, saving over an existing file, reformatting a disk, etc., make sure to confirm the action as shown in the figure titled *Appropriate Confirmation.*

Appropriate Confirmation



Do you want to replace the existing User Feedback.doc?

Confirmation messages are typically worded in the form of a question, they will include a Yes, No, and Cancel button as shown above.

Minimize error messages.

The goal of your program should be to provide enough functionality to make most error messages unnecessary. Presenting the user with too many error messages is usually a sign of the shortcomings of the program. It's better to prevent an error than to report it. To minimize errors:

- *Disable invalid menus, command buttons, and other controls*
- *Provide field defaults whenever possible*
- *Ensure that number fields only accept numbers*
- *Use masks (edit checks) on formatted fields like dates, phone numbers, zip codes, etc.*
- *Prohibit the user from entering an invalid value when a key is pressed*
- *Provide lists of choices whenever possible*
- *Provide file selection dialogs rather than asking users to type filenames*

Provide notification that an error has occurred, an explanation of the error and a solution to fix the errror in all error messages.

Good error messages should provide information that not only tells the user what the problem is, but also, how to fix the problem. Error messages need to contain the following components:

- *Notification that a problem has occurred*
- *Explanation of why the problem occurred*
- *Suggestions on how to fix the problem*

Users do not want to read long and involved error messages. Make sure that you avoid lengthy error messages. Error messages should also be **brief, clear, consistent**, and **specific**.

Error messages will follow these CDC standards:

- *Do not use error numbers*
- *Do not blame the user*
- *Do not use violent terms like fatal, execute, kill, terminate, and abort*
- *Do not use words with all uppercase letters*
- *Do not use exclamation points*
- *Do not use abbreviations*
- *Use full sentences in present or past tense*

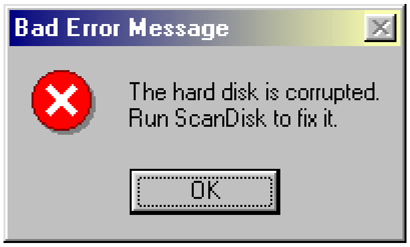
Use the warning (confirmation) icon in field validation error messages.

Errors that are produced as a result of a field being entered incorrectly will use the warning icon. Validation errors can be the result of a single field being in an incorrect format or missing, or the result of cross-field validation between two or more fields.

Do not use an OK command button on a fatal error message.

Bad news is never OK when it comes to fatal errors. Using the OK button in a fatal error message, as illustrated in the figure titled *Bad Error Message*, is a poor choice. By ending the user's interaction with the message box with an OK button, you leave them with the impression that it is okay to have a disk corrupted.

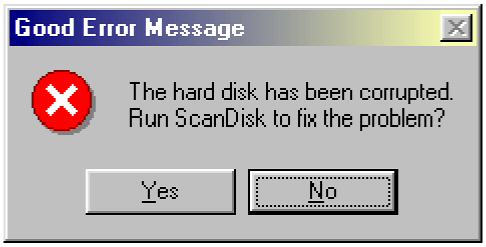
Bad Error Message



The hard disk is corrupted.
Run ScanDisk to fix it.

Instead, report the error by asking a question that helps the user solve the problem as shown in the figure titled *Good Error Message*.

Good Error Message



The hard disk has been corrupted.
Run ScanDisk to fix the problem?

Provide a default button on error messages.

As was stated in the previous standard, it is better to ask questions with a yes or no answer on error messages. In this case, always provide a default button that will respond to the enter key. Make the safest or most frequent option the default button.

User Interaction

Use the wait cursor for operations that take an estimated five seconds or less.

Users need to see feedback that indicates a program is performing some kind of process, even if it is a process that is only taking 5 seconds. By changing the mouse pointer to a wait cursor (typically the hourglass), the user sees that the program will not respond to any further interaction until it has completed the process.

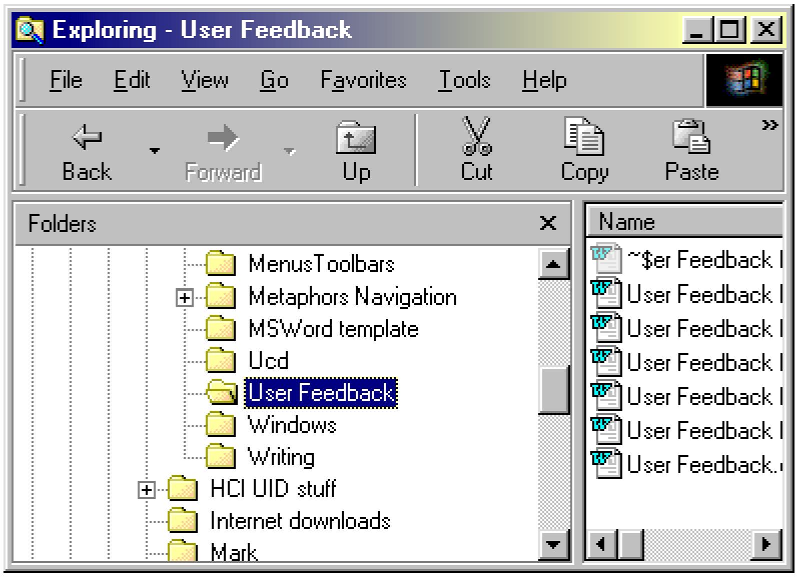
Use progress indicators, and a cancel button option, for operations that take more than an estimated five seconds to perform.

Progress indicators provide more information than a wait cursor by indicating how long the operation will take, how much work has already been done, and that progress is being made. When displaying a progress indicator in a modal dialog box, give the user the ability to stop the operation by providing a cancel button.

Provide visual feedback for selections, such as highlighting, where available in accordance with Windows standards.

When users make explicit selections, the application should display the appropriate selection appearance for each object included in the selection set. This selection appearance is standard for most Windows controls and involves changing the object's background color to the system highlighting color when selected. This can be seen in the figure titled *Selection Feedback*, where the folder that has been selected has had its label background changed to the system highlighting color. In addition some controls, like this tree control, cause the icon associated with the selection to change, as well.

Selection Feedback



Provide immediate visual feedback for non-enterable fields.

In following the rules of visual affordance that were outlined in the Windows section of this style guide, fields that cannot have data entered into them will have a gray background. This provides a visual clue to the user immediately upon viewing the window or screen. If the field is for display only, it should be displayed in a control that is flat with no borders, like a label. For fields that change state based on user interaction, they should be shown in a control with a beveled edge, like a text box. The text box will then be enabled or disabled depending on whether it is enterable at a given time.

Always provide tooltips and title tips.

Tooltips are pop-up windows that provide brief context sensitive help. The user only has to move the mouse cursor over the object and after a brief delay a tooltip appears. Tooltips were originally designed as context sensitive help on toolbars, but their use has been expanded. Tree and list controls should also provide *title tips* to display truncated items within these controls.

Do not use color as the only feedback.

Color can be used as feedback, but is difficult to interpret by itself because color interpretations are culturally dependent, and many users either are color-blind or use monochrome displays. Color is best used to reinforce another form of feedback.

Make use of sound optional, user configurable, and not the only form of feedback.

Sound can provide effective feedback, but it is a transient form of feedback and may be lost. It can be very annoying, especially in a quiet user environment such as an office. It is a poor feedback choice for users who are hearing impaired, or for users who are in a noisy environment. If you use sound, make it optional and user configurable. Sound is best used to reinforce another form of feedback.

General Feedback Guidelines

The following guidelines have been established within the user interface design community and should be applied across CDC software and web-based surveillance applications using HTML.

Messaging

Match message box option buttons to the text.

Message boxes provide a message with options buttons that prompt the user for a response. Do not provide Yes and No buttons to non-questions. Do not provide multiple buttons that have the same effect.

Provide Help buttons in message boxes only when appropriate.

Do not provide a Help button in a message box unless you can provide meaningful help. An example of meaningful help would involve providing the user with an error number and a more detailed description of the error to assist the user in getting technical assistance from a help desk.

Keep status bar messages brief and in the context of the user task.

Status bar messages should be as brief as possible so that the text can easily be read by the user. When writing the status bar message, follow these guidelines:

- *Start messages with a present tense verb (active voice)*
- *Always use terms that are familiar to the user*
- *Avoid technical jargon*
- *When describing a command with a specific function, use words specific to the command*

User Interaction

Provide cursor hinting when appropriate.

Cursor hinting changes the mouse pointer to let the user know that something is currently going on in the application. Listed below are some cursors that should be used:

***Wait Cursor** - hourglass indicating that the operation will take a few seconds*
***Not Available Cursor** - international no symbol indicating that the object being pointed at is not available as a drop target. This can also apply to showing displayable fields as being non-editable.*
***Zoom Cursor** - magnifying glass indicating that the current view is being zoomed on.*
***Hand Cursor** - a hand or glove indicating a hyperlink is present.*

There are many other common pointer shapes that provide feedback about a particular location, operation, or state. For a complete list of common pointers, see Microsoft's *Windows Interface Guidelines for Software Design*.

Provide scroll bar tips and data tips when appropriate.

Tooltips were originally designed as context sensitive help on toolbars, but their use has been expanded. Use these tooltips when appropriate:

- ***Scroll bar tips** - to display scroll bar location*
- ***Data tips** - to display more information about a control*

Provide mouseovers when appropriate.

Like rollovers on the web, mouseovers provide immediate feedback to users when the mouse cursor is moved over a screen element. Mouseovers can result in the visual look of the element changing, or the display of a message or supplemental information in the status bar. Too much feedback can be distracting; therefore, mouseovers should be used with care. A good use of a mouseover would be to display an instructional message in the status bar to aid the user in entering data into a specific field.

HTML Specific Feedback Standards

The following standards are to be adopted across all CDC web-based applications using HTML. These standards are intended to lead to a more consistent and usable interface.

Always use the ALT attribute of the <IMG> tag.

The text in the ALT attribute of the <IMG> tag in HTML displays on a Web page even when the user opts to turn the display of graphics off. A visually impaired user may employ a special device that translates the Web page into electrical impulses that they can understand. The text in the ALT attribute can supply valuable information to all those users. The text in the ALT attribute can also provide additional information that will help all users decide whether to click on a graphical link or not.

Always use client-side image maps instead of server-side image maps.

Some Web sites use image maps as navigational devices. The image map is a single graphic where multiple, discrete areas (hot areas) in the image are designated as links through the use of x,y coordinates within the image map. Image maps can be implemented either on the client side in HTML, or on the server side in code.

When a user moves the cursor over a hot area in a client-side image map, the URL of the link appears in the bottom left of the browser window. When the user movers the cursor over a hot area in a server-side image map, the x,y coordinates of the hot area appear in the bottom left of the browser window. The URL is meaningful to the user, x,y coordinates are not. If you are going to use an image map, use a client-side image map, not a server-side image map.

Always provide user feedback for actions that initiate server-side processing.

When a user completes a form, requests a search, modifies data, or initiates any action that results in server-side processing, always provide feedback as to the success or failure of the action.

For example, there are two common types of feedback when a user completes a form then clicks on the Submit button:

1. Confirmation that the form was accepted as completed successfully
2. A message that the form was not accepted and what action the user should take to resubmit the form and have it accepted

In both cases, validation of the form takes place on the server, the appropriate response page is created on the server, then the page is sent to the user and displayed in the user's browser.

Support and provide access to plug-ins used by CDC web-based surveillance applications.

Plug-ins can add much value to user interaction (including user feedback) if they download quickly and support the purpose of the web application or web site. The code for some plug-ins is already included in most browsers while some is not. Make sure to provide access to and support any plug-ins that are used within CDC surveillance application.

<u>HTML Specific Feedback Guidelines</u>

The following guidelines are recommended and should be applied across CDC web-based surveillance applications using HTML.

There are a number of types of user feedback on the Web that users have come to expect. The table titled *Types of Web Feedback* describes each type.

Types of Web Feedback

Initiating Action	Resulting User Feedback	How Feedback Is Created
The user clicks on a button or link that causes information to be retrieved (downloaded) from the server.	Icon in upper right corner of browser becomes animated. The status bar in lower left of browser indicates progress of download.	Automatically done by browser.
The user passes the cursor over a textual link.	Cursor changes to a hand with pointing index finger. The status bar in lower left of browser shows URL of link.	Automatically done by browser.
The user passes the cursor over a visual link.	Cursor changes to a hand with pointing index finger. The status bar in lower left of browser shows URL of link.	Automatically done by browser.
	Optional; A small box with text appears near the link.	The developer must supply text in the ALT attribute of the <IMG> tag in HTML
	Optional; Image changes in some way (highlights, new image appears, text appears somewhere else on screen, etc.). This is a **rollover**.	The developer must supply another image that, through use of JavaScript or other code, replaces a portion of the screen, giving the illusion of a rollover.

The user clicks on a visual or textual link that causes a plug-in to be activated.	Some type of interaction occurs. Animation begins, sound plays, the screen appears to jump to the right, many different actions may occur. This is an indication that a **plug-in** has been activated. The interaction may be as simple as highlighting a graphic or as complex as playing a game.	The developer embeds a link (using the <EMBED> tag in HTML) to a file created using software (such as Director, Flash, or Authorware). Special code must be added (or plugged in) to the browser software to execute the file.
The user clicks on a button, such as **Search**, **Submit**, **Add**, **Delete**, etc.	A new page appears with user feedback containing the results of the action implied by the button. The user feedback can be a **confirmation** that the action was completed successfully, or an **error message** that the action was not completed successfully, or some other message to the user.	Control passes to a server-side program that performs the action implied by the button title. The server-side code creates the user feedback and sends it to the client to be displayed as a new page in the browser.

Consider the use of rollovers in Web navigation.

Rollovers provide immediate feedback to users when they move the cursor over a graphical link. Rollovers are a stronger visual indication of graphical links than the cursor changing to a hand. Rollovers are easier for people to see. Rollovers can also supply additional information that helps the user decide whether to click on the link or not.

Recommended Readings

Cooper, Alan. *About Face: The Essentials of User Interface Design*. IDG Books Worldwide, Inc., 1995
Chapter 15 - Elephants, Mice and Minnies

Horton, William. Designing & Writing Online Documentation.

McKay, Everett N. *Developing User Interfaces for Microsoft Windows*. Microsoft Press, 1999
Chapter 4 - Establishing a Consistent User Interface Style
Chapter 10 - Good Interfaces are Invisible
Chapter 12 - Learn from The Design of Everyday Things
Chapter 13 - Learn from the Web

Chapter 23 - Unnecessary Message Boxes Are Pure Evil
Chapter 26 - Programmer Testing
Chapter 30 - Check Your Error Messages

Fleming, Jennifer, *Web Navigation for the World Wide Web*, O'Reilly & Associates, Inc., 1998

Morville & Rosenfeld. *Information Architecture for the World Wide Web*, O'Reilly & Associates, Inc., 1998
The Windows Interface Guidelines for Software Design. Microsoft Press, 1995
Chapter 1 - Design Principles and Methodology
Chapter 4 - Input Basics
Chapter 5 - General Interaction Techniques
Chapter 8 - Secondary Windows
Chapter 12 - User Assistance
Chapter 14 - Special Design Considerations

Helpful Web Sites

The Microsoft Developers Network Online Library of Books, specifically the online version of *The Windows Interface Guidelines for Software Design*
http://msdn.microsoft.com/isapi/msdnlib.idc?theURL=/library/books/winguide/PLATFRM2/D5/S115B5.HTM

Isys Architects Interface Hall of Shame contains a very good article on writing good error messages
http://www.iarchitect.com/mshame.htm